ATCA Test Card User Manual

Revision History:
Rev 1.0 09.08.2013 First Draft
Rev 1.1 12.09.2013 Added Chapter E-keying

Key features

- ***Evaluation Board for the ATCA IPMC***
- ***Implements all optional features of the IPMC: Analog inputs for voltage, current or temperature measurements, external I2C temperature sensors, external thermal-diode sensors, tachometer sensors***
- ***Hot swap switch***
- ***Blue Led, Led 1, Led 2, Led 3***
- ***Payload interfacing signals***
- ***Reference schematic***
- ***GUI software for configuration***
- ***All signals are available to the user through different connectors***

Illustration 1: ATCA Test Card

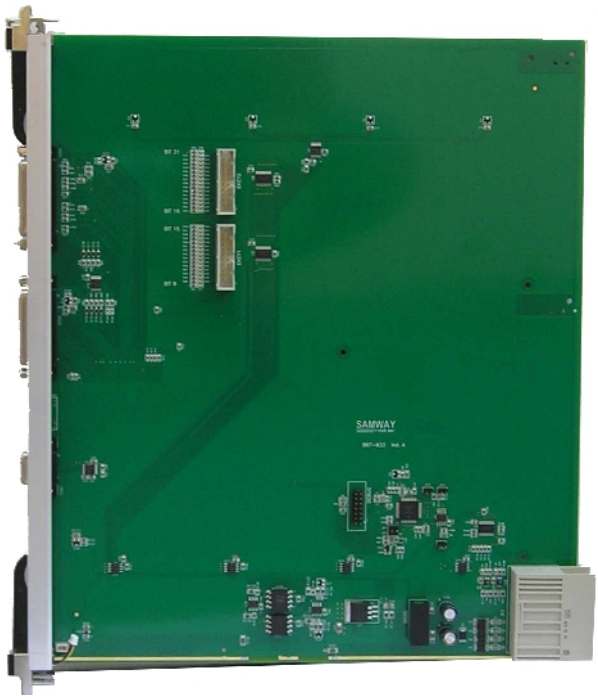

Table of Contents

Key features..... 1
Description..... 2
Sensors..... 3
Configuring Sensors..... 5
FRU Information..... 8
E-keying 9
Payload available signals..... 11
Leds:..... 11
Command Line Interface (CLI) 12
List of CLI commands..... 12
1 help command 12
2 reboot command 12
3 sensor command 12
4 uptime command 13
5 version command 13
6 xmodem command 13

Updating the FRU and SDR files..... 14
Connectors:..... 15

Illustration Index

Illustration 1: ATCA Test Card..... 1
Illustration 2: IPM Controller..... 2
Illustration 3: SDR Compiler..... 5
Illustration 4: Selecting only the necesary sensors..... 6
Illustration 5: Window for changing the parameters of a analog sensor..... 6
Illustration 6: Various embedded options for defining the SDR raw formula..... 7
Illustration 7: FRU File Compiler..... 8
Illustration 8: Samway OEM Ekey Add-On Record..... 9
Illustration 9: Tera Term Screen shot for sending a file using xmodem..... 14
Illustration 10: Connectors Position..... 15

Index of Tables

Table 1: Supported Set of Sensors..... 4
Table 2: Payload available Signals..... 11
Table 3: Available Leds..... 11
Table 4: Voltage and Current Requirements..... 15
Table 5: DSUB1 Connector pinout..... 16
Table 6: DSUB2 Connector pinout..... 17
Table 7: EKEY1 Connector Pinout..... 18
Table 8: EKEY2 Connector Pinout..... 19

Description

The ATCA Test Card is intended to be an evaluation board for the ATCA IPM Controller. The board implements all the IPMC features available, and allows rapid product development by providing easy access to all IPMC signals.

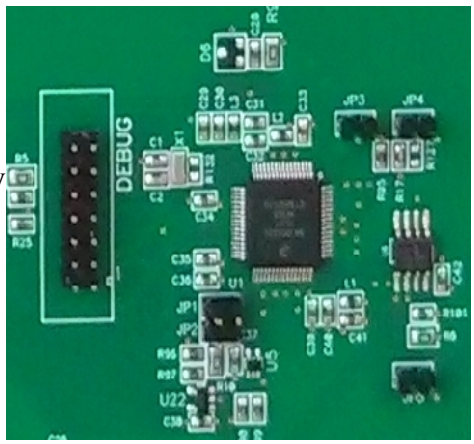

Illustration 2: IPM

All signal are available on a series of connectors. This way any prototype board could be easily connected to the IPMC without the effort of building a dedicated PCB.

The IPMC software is pre-programmed on Freescale K10DX256 microcontrollers and provides all IPMI functionality required by the PICMG 3.0. Besides the microcontroller the IPMC circuit only uses a small set of additional components in order to fulfill the mandatory requirements of the specs. For implementing optional features additional circuits are required.

The ATCA Test card implements all additional circuits required by the IPMC in order to provide all the monitoring and control features.

Sensors

The IPMI Controller supports a limited, fixed, set of sensors. Each supported sensor has a predefined sensor number and if it is used,it shall have a Sensor Data Record (SDR) associated to it. Using the set of SDRs the IPMI software knows what sensors are implemented by the card and how to monitor them.

The ATCA Test Card implements the complete sensor set. The supported sensor list is defined by the following table.

Sensor Number	Default Sensor Name	Description
1	**Hot Swap**	**Discrete sensor**: offers information about the hot swap state of the IPMC 0 = M0 – FRU Absent 1 = M1 - FRU Inactive 2 = M2 – FRU Activation Request 3 = M3 – FRU Activation In Progress 4 = M4 – FRU Active 5 = M5 – FRU Deactivation Request 6 = M6 – FRU Deactivation In Progress 7 = M7 – FRU Communication Lost
2	**Hot Swap Handle**	**Discrete sensor**: offers information about the hot swap handle: 0 = Handle Open 1 = Handle Closed
3	**IPMB0_Status**	**Discrete sensor**: offers information about the state of the IPMB0 bus: IPMB_A ok, error, enabled, disabled IPMB_B ok, error, enabled, disabled
4	**An0**[1]	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal
5	**An1**	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal
6	**An2**	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal
7	**An3**	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal
8	**An4**	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal
9	**An5**	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal
10	**An6**	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal

1 The analog input pins withstand signals in the range 0..2.5 Volts. If the signals that need to be monitored are outside this range, additional circuitry will be necessary (voltage dividers). For examples of circuits used in monitoring common voltages (3.3V, 12V, -12V, -48V) please refer to the reference schematic

Sensor Number	Default Sensor Name	Description
11	**An7**	**Analog Sensor:** can be used to monitor a voltage, current or temperature signal
12	**Temp1**	**Analog Sensor**: monitors a **TMP75** I2C temperature sensor configured for the I2C address **0x90**
13	**Temp2**	**Analog Sensor**: monitors a **TMP75** I2C temperature sensor configured for the I2C address **0x92**
14	**Temp3**	**Analog Sensor**: monitors a **TMP75** I2C temperature sensor configured for the I2C address **0x94**
15	**Temp4**	**Analog Sensor**: monitors a **TMP75** I2C temperature sensor configured for the I2C address **0x98**
16	**Max6699_Temp1**	**Analog Sensor**: monitors a thermal diode connected to input 1 of the **Max6699** circuit
17	**Max6699_Temp2**	**Analog Sensor**: monitors a thermal diode connected to input 2 of the **Max6699** circuit
18	**Max6699_Temp3**	**Analog Sensor**: monitors a thermal diode connected to input 3 of the **Max6699** circuit
19	**Max6699_Temp4**	**Analog Sensor**: monitors a thermal diode connected to input 4 of the **Max6699** circuit
20	**Max6699_Local_Sensor**	**Analog Sensor**: monitors the board temperature sensed by the **Max6699** circuit
21	**FAN1**	**Analog Sensor**: monitors Tachometer input 1
22	**FAN2**	**Analog Sensor**: monitors Tachometer input 2
23	**FAN3**	**Analog Sensor**: monitors Tachometer input 3
24	**FAN4**	**Analog Sensor**: monitors Tachometer input 4
25	**FAN5**	**Analog Sensor**: monitors Tachometer input 5
26	**FAN6**	**Analog Sensor**: monitors Tachometer input 6
27	**FAN7**	**Analog Sensor**: monitors Tachometer input 7
28	**FAN8**	**Analog Sensor**: monitors Tachometer input 8

Table 1: Supported Set of Sensors

Configuring Sensors

The IPMC uses standard, IPMI compliant SDR records in order to monitor the board parameters.

The SDR repository of a board will be a software image of the hardware sensors. So in order to allow a quick and simple set-up, a GUI SDR compiler is provided.

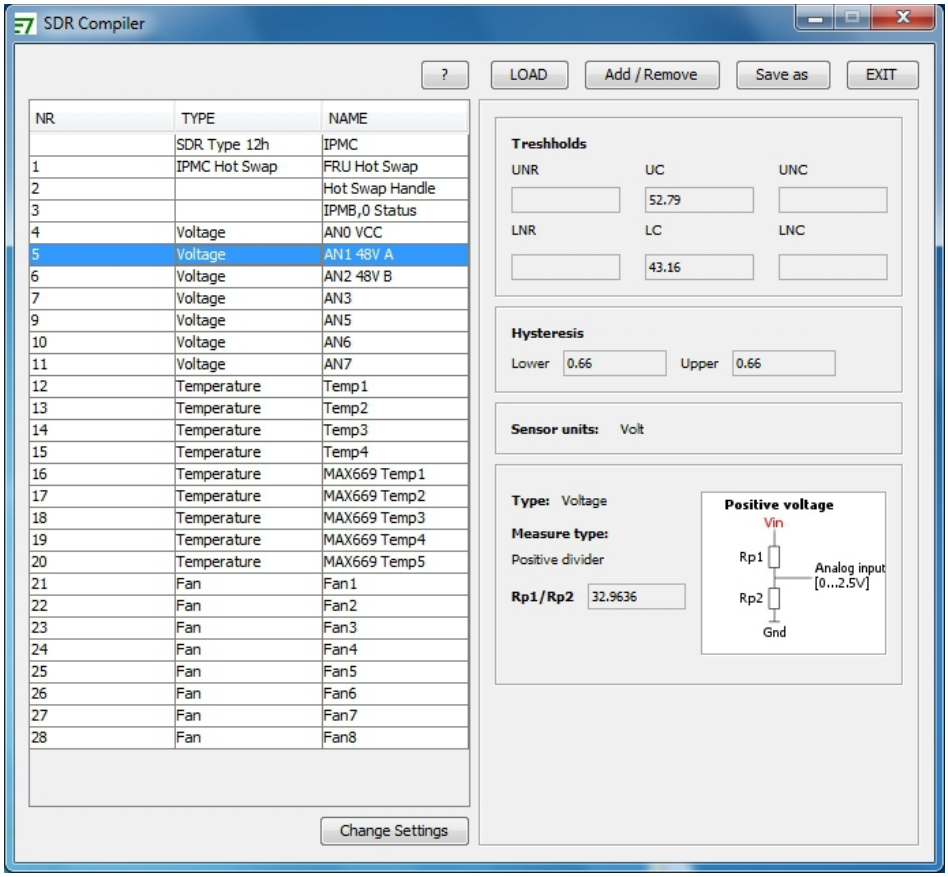

Illustration 3: SDR Compiler

Using the GUI compiler a subset of all of the supported sensors can be defined by a simple select operation.

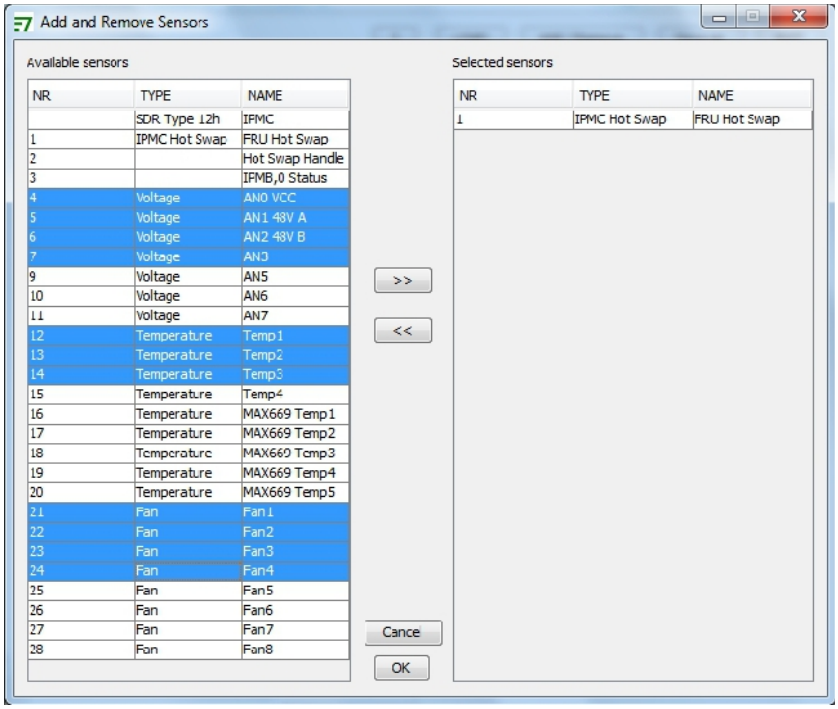

Illustration 4: Selecting only the necesary sensors

After the SDR set has been defined, all the sensors can be customized:

- threshold and hysteresis values can be changed for analog sensors
- names can be changed for all sensors

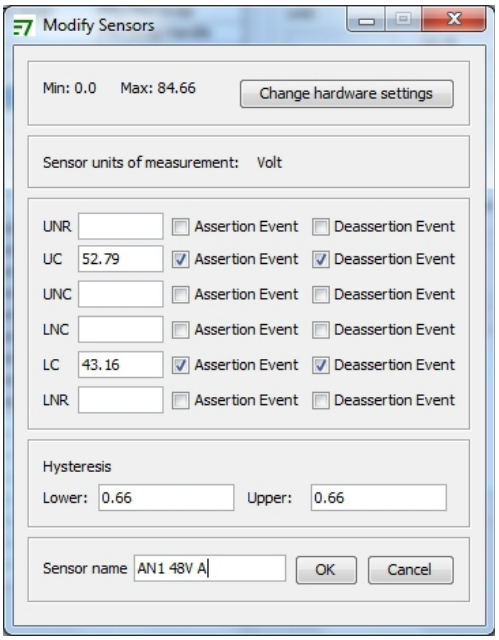

Illustration 5: Window for changing the parameters of a analog sensor

- for the analog inputs the raw SDR formulas of a few hardware circuits have been implemented. This feature allows an easy integration of common analog set-ups: positive voltage divider, negative voltage divider, and gain block. For these common circuits only the divider/gain value has to be inputed, and the raw conversion formula will be automatically computed by the software. For more complex circuits, the raw formula can be inputed manually.

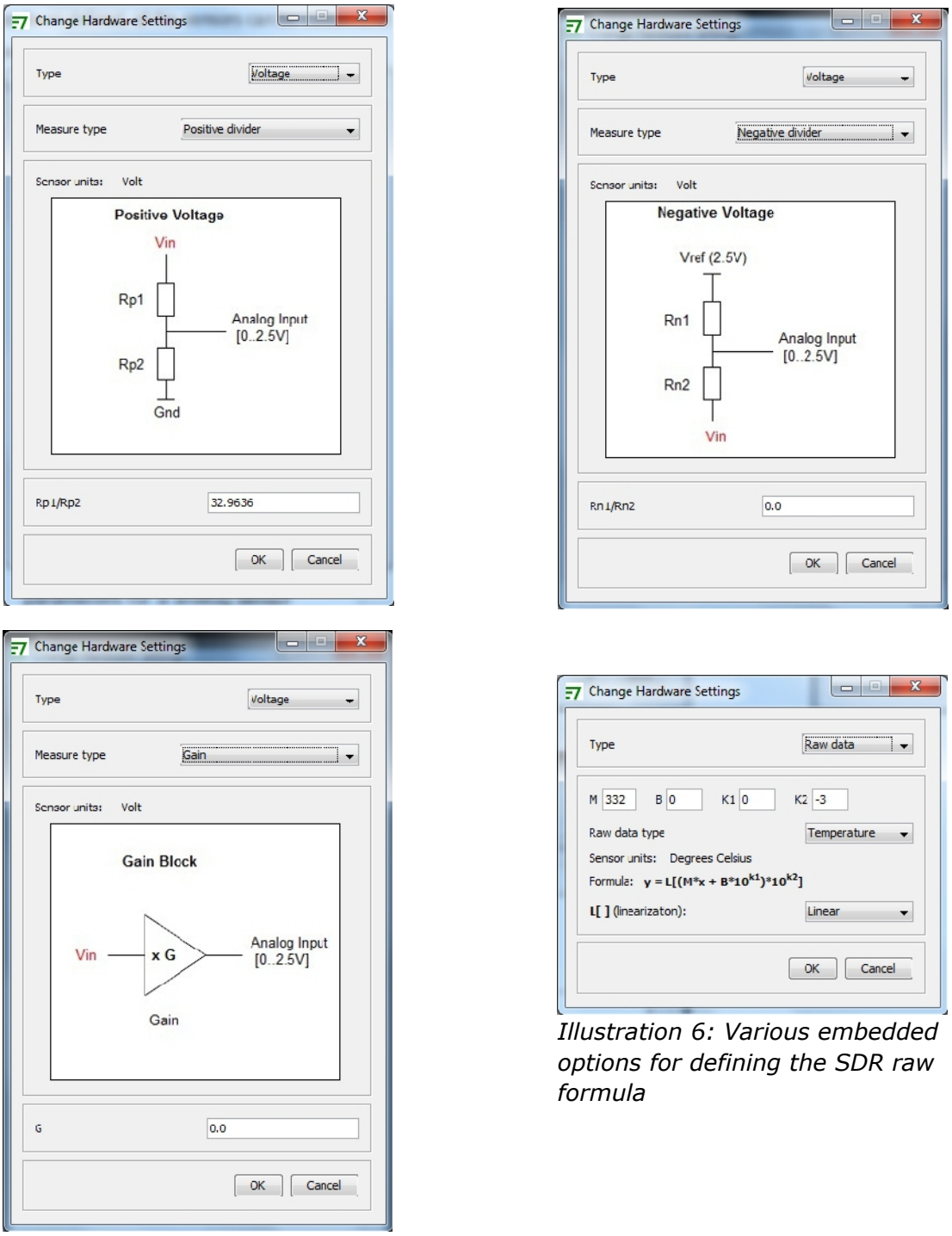

Illustration 6: Various embedded options for defining the SDR raw formula

FRU Information

The IPMC controller will be used in an IPMI environment. In order to interact to the other FRUs in the system, the board will have to host a FRU information file. This type of file contains important information concerning the board:

- Manufacturer's name
- Part Number
- Serial Number
- Revision
- Manufacturing date and time
- Information related to the communication protocols implemented over the Base, Fabric, Timing and Local Bus interfaces
- other IPMI related information

All the required information can be saved in the FRU file format using the GUI FRU compiler.

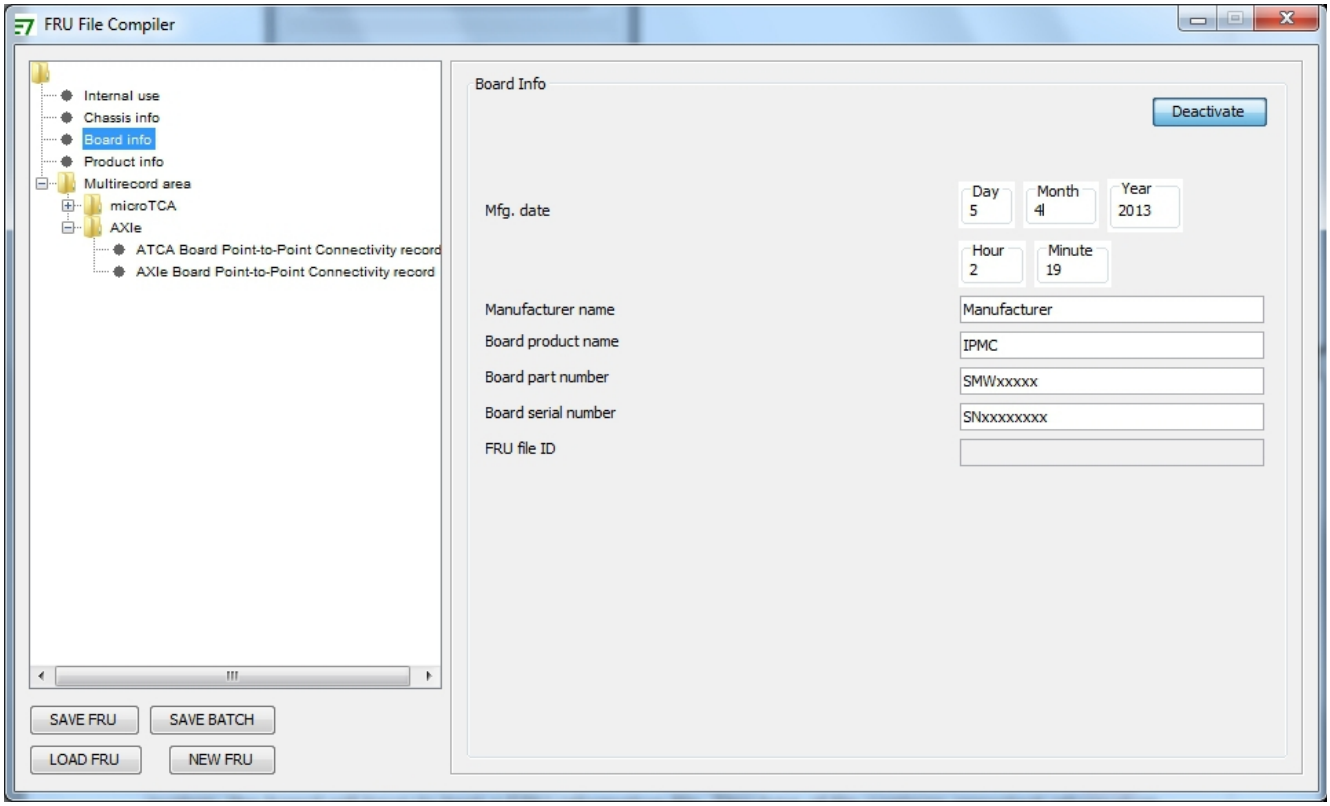

Illustration 7: FRU File Compiler

E-keying

For E-Keing the IPMC software implements 2 mechanisms:

- Forwarding the data received from the Shelf Manager via the “Set Port state” command(The command data is forwarded to a user defined I2C address)
- 2 user configurable 16 bit IO-Expanders. The IO-Expanders allow up to 31 user configurable pins that can act as active high outputs and can be either asserted or deasserted for either enabling or disabling the link.

Either one, both or none of the above options can be used for implementing E-keying.
In order to allow a simple setup the GUI FRU Compiler implements Samway's OEM Ekey Add-On record.

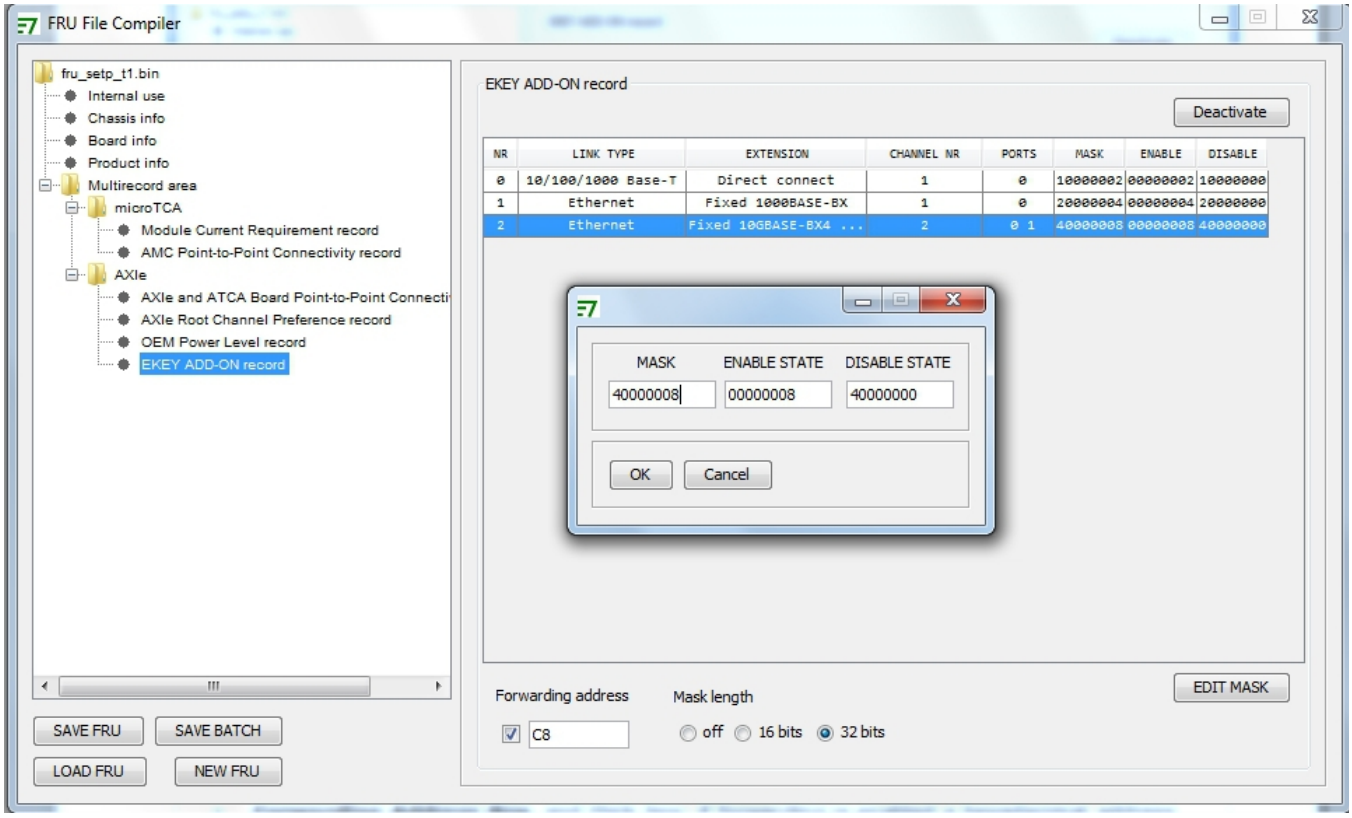

Illustration 8: Samway OEM Ekey Add-On Record

For every link the 32 IO pins behavior is defined by a set of tree hexadecimal values:

- **Used Pins Mask**: the mask defines the pins that will be affected by a state change for the link. If a bit's value is 1, the corresponding pin is used by the current link.
- **Enable Link State:** the mask defines the values for the used bits when the link is enabled. If a bit's value is 1 the pin will be asserted, otherwise it will be deasserted.
- **Disable Link State:** the mask defines the values for the used bits when the link is disabled. If a bit's value is 1 the pin will be asserted, otherwise it will be deasserted.

The tree masks are hexadecimal values that represent the 32 IO Expander pins: Ekey0 ..Ekey31. Ekey0 is reserved for the MASK_VALID signal and is not available to the user. All the other pins can be used freely.

For the values in Illustration 8:

- **Used Pins Mask**: for the current link, pins Ekey3 and Ekey30 are used
- **Enable Link State**: when the link is enabled Ekey3 is asserted and Ekey30 is deasserted
- **Disable Link State**: when the link is disabled Ekey3 is deasserted and Ekey30 is asserted

IPMC E-keying procedure:

1. The IPMC receives the "Set Port State" command from the shelf manager
2. If forwarding is enabled and a non-zero I2C address has been defined, the command data is forwarded on the external I2C bus using the following format:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4..7	Byte 8
I2C Address	0x01 **(FWD command)**	0x00	**Link Type**	**Link Descriptor**	**State**

Link Type : 0x02 ATCA link
0x04 AXIe link
all other values are reserved

Link Descriptor: 4 byte value per ATCA/ AXIe received from the Shelf Manager

State : 0x00 Disable
0x01 Enable
all other values are reserved

3. If the IO Expanders are used and the tree masks have been defined for the current link, the MASK_VALID signal is deasserted.
4. The values for the pins defined by the **Used Pins Mask** are changed. If the link is enabled the pin values are defined by the **Enable Link State** mask, and if the link is disabled the values are changed according to the **Disable Link State** mask.
5. After the pin values have been changed, the MASK_VALID signal is asserted.
6. The IPMC sends the completion code to the shelf manager for the E-keying command.

Payload available signals

The IPMC is supposed to represent the payload over the IPMI. In order to do so, a limited set of signals that form a simple interface between the payload and the IPMC have to be implemented. The IPMC implements the following signals:

Signal Name	Type	Active level	Description
PP_RST	Output	High	Payload reset signal
SHDN_RDY#	Input	Low	When the signal is active the payload is ready to be shutdown
Payload_SCL	In-Out		I2C serial clock
Payload_SDA	In-Out		I2C serial data
PCIe_Ready#	Input	Low	When the signal is active the payload has finished the PCIe initialization
GPIO[3:1]	In-Out		3 General Purpose pins
SLOT_ID[4:1]	IO Expander Output		4 signals that represent the slot number for the current board position in the system
Temp_Fail#	IO Expander Output	Low	If any of the temperature sensor violates any of their thresholds, this signal is asserted
SHDN_REQ#	IO Expander Output	Low	When active, signals the payload that a shutdown has been requested
PP_EN#	IO Expander Output	Low	Enable for the Payload Power switch
LED2	In-Out		Green Led that can be controlled both by the payload and the IPMC
LED3	In-out		Amber Led that can be controlled both by the payload and the IPMC

Table 2: Payload available Signals

Leds:

Name	Color	Control	Description
Blue Led	Blue	IPMC	Hot swap led
Led 1	Green,Red, Amber	IPMC	Status Led: Green – ok Red - error
Led 2	Green	IPMC, Payload	User defined
Led 3	Amber	IPMC, Payload	User defined

Table 3: Available Leds

Command Line Interface (CLI)

The ATCA Test card provides a RS232 serial interface through which the commands of the Command Line Interface (CLI) can be sent.

On Windows systems, we recommend the use of “Tera Term” or “Hyperterminal” as the terminal programs.

The CLI is accesible on the COM1 connector using a 1:1 serial cable for a direct connection to the serial port of a PC

Terminal settings:

- 115200 bits per second
- data bits: 8
- parity: none
- stop bit: 1

For file transfer the CLI implements the xmodem protocol.

List of CLI commands

1 help command

Syntax: help
Function: Displays a list of the available commands.

2 reboot command

Syntax: reboot
Function: Restarts the IPMC
Example 1:

```
%>reboot
 System will restart! Please wait...
```

3 sensor command

Syntax: sensor
Function: Displays information for the installed set of sensors.
Example 1:

```
%>sensor
---------------------Sensor List-----------------------------
--no--Name-------------Value--Unit---State-----------------
*   1  FRU Hot Swap     M0: FRU Not Installed
*   2  Hot Swap Handle
*   3  IPMB-0 Status    IPMB A: ok   , Enabled
                        IPMB B: ok   , Enabled
*   4  AN0 VCC          3.31   V      Ok
*   5  AN1 48V A        47.14  V      Ok
```

```
*   6  AN2 48V B        47.48  V      Ok
*   7  AN3              0.01   V      Ok
*   8  AN4              0.01   V      Ok
*   9  AN5              0.00   V      Ok
*  10  AN6              0.01   V      Ok
*  11  AN7              0.01   V      Ok
*  12  Temp1            28.00  deg C  Ok
*  13  Temp2            28.00  deg C  Ok
*  14  Temp3            27.00  deg C  Ok
*  15  Temp4            28.00  deg C  Ok
*  16  MAX669 Temp1     28.00  deg C  Ok
*  17  MAX669 Temp2     28.00  deg C  Ok
*  18  MAX669 Temp3     27.00  deg C  Ok
*  19  MAX669 Temp4     28.00  deg C  Ok
*  20  MAX669 Temp5     28.00  deg C  Ok
*  21  Fan1             0      RPM    LC
```

4 uptime command

Syntax: uptime
Function: Displays the amount of time which has past since the IPMC became operational.
Example 1:

```
%>uptime
 Uptime=0 days 03:05:12
```

5 version command

Syntax: version
Function: Displays various information about the IPMC: firmware version, Hardware Id, Slot ID.
Example1:

```
%>version
 IPMC FW 1.1
 Hardware Id : 7
 Slot : 3
```

6 xmodem command

Syntax: xmodem fru | sdr
Function: Upload the FRU or SDR file to the IPMC using the xmodem protocol
Example 1:
%>xmodem fru
Please upload the file...
%>...Done!

Updating the FRU and SDR files

In order to configure the IPMC of the ATCA test card two files are required: the FRU and SDR file. Both can be easily created using the GUI software suites that accompany the IPMC: FRU File compiler and SDR File compiler.

Creating new files or modifying old ones is really straight forward due to the graphical interface. For more details on all the available options please refer to the respective software user manuals.

After the files are created they have to be uploaded using the CLI available on COM1.

For uploading a file the following steps are required :

1. Connect to the CLI interface
2. Issue the **xmodem** command, using the correct parameter:

%> xmodem fru | sdr

3. Upload the file using the terminal program
4. After the file transfer is completed a confirmation message will be displayed. At this point the file has been saved and a reboot is required in order to activate the changes.

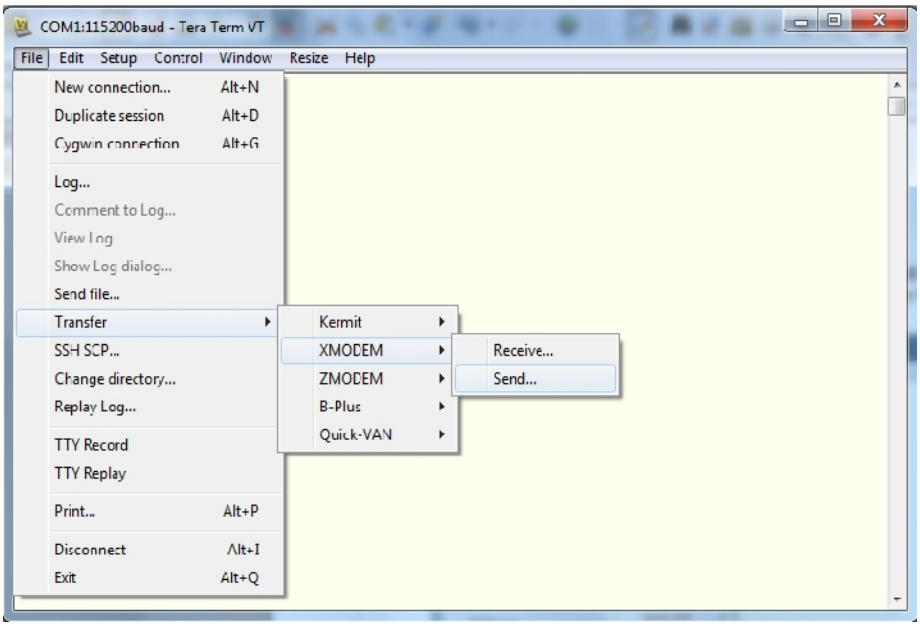

Illustration 9: Tera Term Screen shot for sending a file using xmodem

Connectors:

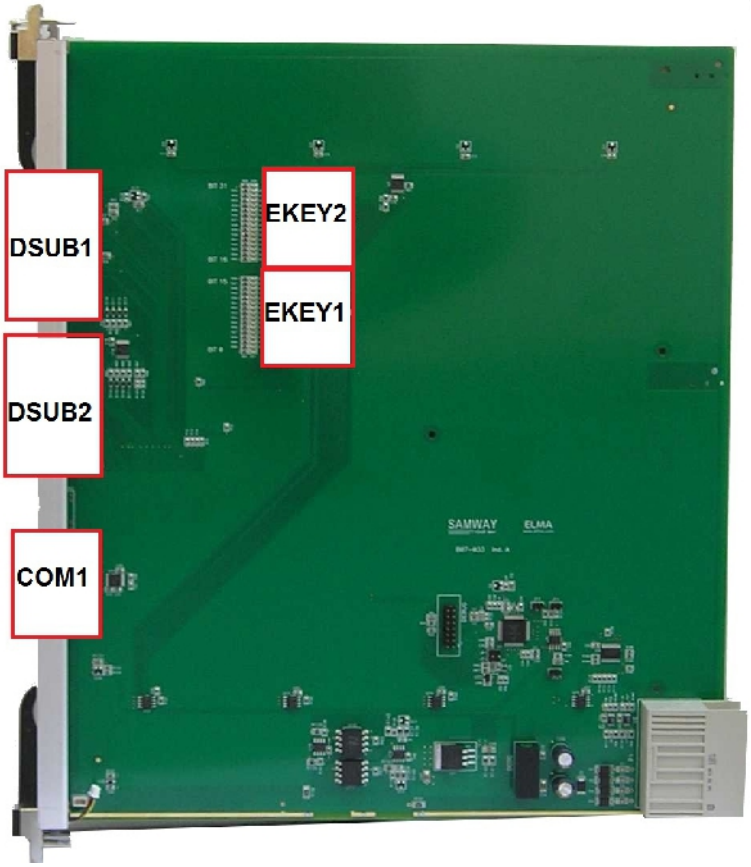

Illustration 10: Connectors Position

The ATCA Test card uses a set of connectors in order to provide access to all the available signals.

Voltage and current maximum ratings

Simbol	Description	Min.	Max.	Unit
V_{digIn}	Digital Pin Input Voltage	-0.3	+5.5	V
I_{digIn}	Digital Pin Current	-25	+25	mA
V_{digOut}	Digital Pin Output Voltage	0	Vcc (+3.3)	V
V_{anIn}	Analog Pin Input Voltage	-0.3	Vcc+0.3(+3.6)	V
V_{adcMax}	ADC input Full Scale voltage		Vref (+2.5)	V

Table 4: Voltage and Current Requirements

No.	Name	Type	Obs.
1	PCIE_READY#	Digital Input	10 kΩ pull up
2	PP_RST	Digital Output	10 kΩ pull up
3	PP_EN#	Digital Output	10 kΩ pull up
4	GPIO1	Digital In-Out	10 kΩ pull up
5	GPIO2	Digital In-Out	10 kΩ pull up
6	GPIO3	Digital In-Out	10 kΩ pull up
7	NC		
8	SHDN_RDY#	Digital Input	10 kΩ pull up, JP1 – GND override
9	SLOT_ID1	Digital Output	10 kΩ pull up
10	SLOT_ID2	Digital Output	10 kΩ pull up
11	SLOT_ID3	Digital Output	10 kΩ pull up
12	SLOT_ID4	Digital Output	10 kΩ pull up
13	LATCH_IO5	Digital In-Out	10 kΩ pull up
14	+3.3V	Analog OUT	
15	TACH_IN1	Digital Input	10 kΩ pull up, FAN1 input
16	GND		
17	TACH_IN2	Digital Input	10 kΩ pull up, FAN2 input
18	GND		
19	TACH_IN3	Digital Input	10 kΩ pull up, FAN3 input
20	GND		
21	+3.3V	Analog OUT	
22	TACH_IN4	Digital Input	10 kΩ pull up, FAN4 input
23	GND		
24	EXT_P_OUT1	Analog OUT	Open collector PWM signal. External Pull-up required. R128 (10 kΩ Pull-up to 3.3V) can be assembled
25	SHDN_REQ#	Digital Output	10 kΩ pull up

Table 5: DSUB1 Connector pinout

No.	Name	Type	Obs.
1	AN0	Analog Input	Already Used. Monitors internal +3.3V rail.
2	AN1	Analog Input	Already Used. Monitors internal -48V A rail.
3	AN2	Analog Input	Already Used. Monitors internal -48V B rail.
4	AN3	Analog Input	1 kΩ pull down
5	AN4	Analog Input	1 kΩ pull down
6	AN5	Analog Input	1 kΩ pull down
7	AN6	Analog Input	1 kΩ pull down
8	AN7	Analog Input	1 kΩ pull down
9	TEMP_FAIL#	Digital Output	10 kΩ pull up
10	PAYLOAD_SDA	Digital In-Out	Serial Data pin for Payload I2c bus. 2.7 kΩ pull up
11	PAYLOAD_SCL	Digital In-Out	Serial Clock pin for Payload I2c bus. 2.7 kΩ pull up
12	TXD0	Digital Output	Transmit pin of RS232 interface
13	RXD0	Digital Input	Receive pin of RS232 interface
14	+3.3V	Analog Output	
15	TACH_IN5	Digital Input	10 kΩ pull up, FAN5 input
16	GND		
17	TACH_IN6	Digital Input	10 kΩ pull up, FAN6 input
18	GND		
19	TACH_IN7	Digital Input	10 kΩ pull up, FAN7 input
20	GND		
21	+3.3V	Analog Output	
22	TACH_IN8	Digital Input	10 kΩ pull up, FAN8 input
23	GND		
24	EXT_P_OUT2	Analog OUT	Open collector PWM signal. External Pull-up required. R129 (10 kΩ Pull-up to 3.3V) can be assembled
25	GND		

Table 6: DSUB2 Connector pinout

No.	Name	Type
1	EKEY0	Digital Output
2	EKEY1	Digital Output
3	EKEY2	Digital Output
4	+3.3V	
5	EKEY3	Digital Output
6	EKEY4	Digital Output
7	GND	
8	EKEY5	Digital Output
9	EKEY6	Digital Output
10	EKEY7	Digital Output
11	EKEY8	Digital Output
12	+3.3V	
13	EKEY9	Digital Output
14	EKEY10	Digital Output
15	GND	
16	EKEY11	Digital Output
17	EKEY12	Digital Output
18	EKEY13	Digital Output
19	EKEY14	Digital Output
20	EKEY15	Digital Output

Table 7: EKEY1 Connector Pinout

No.	Name	Type
1	EKEY16	Digital Output
2	EKEY17	Digital Output
3	EKEY18	Digital Output
4	+3.3V	
5	EKEY19	Digital Output
6	EKEY20	Digital Output
7	GND	
8	EKEY21	Digital Output
9	EKEY22	Digital Output
10	EKEY23	Digital Output
11	EKEY24	Digital Output
12	+3.3V	
13	EKEY25	Digital Output
14	EKEY26	Digital Output
15	GND	
16	EKEY27	Digital Output
17	EKEY28	Digital Output
18	EKEY29	Digital Output
19	EKEY30	Digital Output
20	EKEY31	Digital Output

Table 8: EKEY2 Connector Pinout